Plotting package evaluation

Introduction

We would like to evaluate several graphics packages for possible use in the GLAST Standard Analysis Environment. It is hoped that this testing will lead to a recommendation for a plotting package to be adopted for use by the science tools. We will describe the packages we want to test, the tests we want to do to (given the short time and resources for doing this), and then the results of the evaluation.

Finally we will discuss the conclusions of our testing and hopefully make a recommendation.

According to the draft requirements document for plotting packages the top candidates are:

ROOT
VTK
VisAD
JAS
PLPLOT

There has been some discussion about using some python plotting package: e.g.,Chaco, SciPy, and possibly Biggles (suggested in Computers in Science and Engineering).

A desired feature is have is the ability to get the cursor position back from the graphics package. We will look for this desired feature.

An additional desired feature would be to have the same graphics package make widgets or have a closely associated widget friend. Widget friends will not be tested here, but will have to studied before agreeing to use it.

Package	Widget Friend(s)	Comments
Biggles	WxPython	
Plplot	PyQt, Tk, java	The Python Qt interface is only experimental at present.
ROOT	Comes with its own GUI	INTEGRAL makes GUIs from ROOT graphics libs. We hear this was a bit of a challenge to do, but much of the work is already done

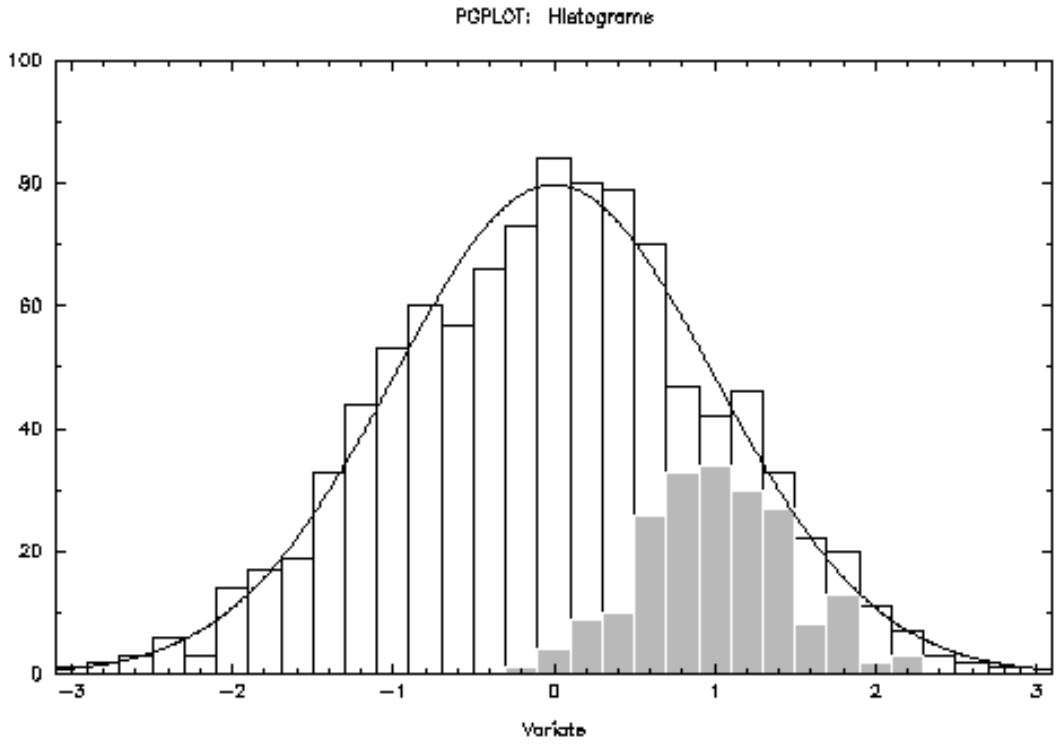
		for us.

Tests:

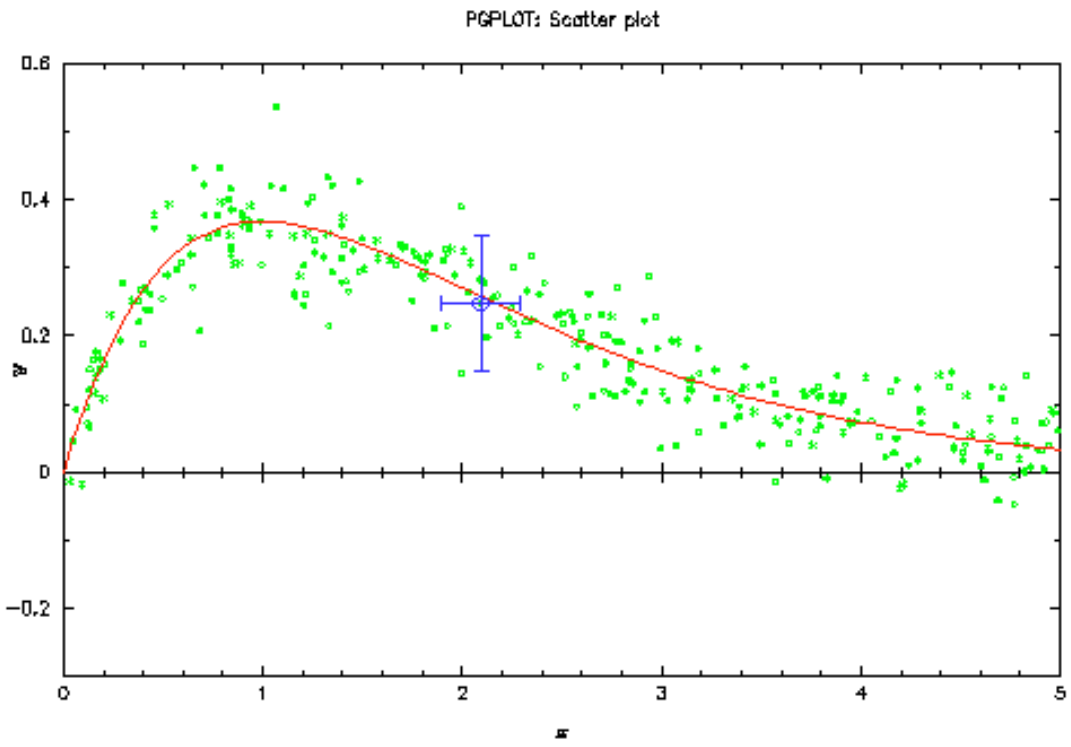
The testing is to be carried out separately in the Windows and Linux environments. For Linux we use Red Hat 7.2, and for Windows we use Windows 2000. The following tests and questions were applied to each package:

Create examples of the following 3 plots taken from the PGPLOT web page:

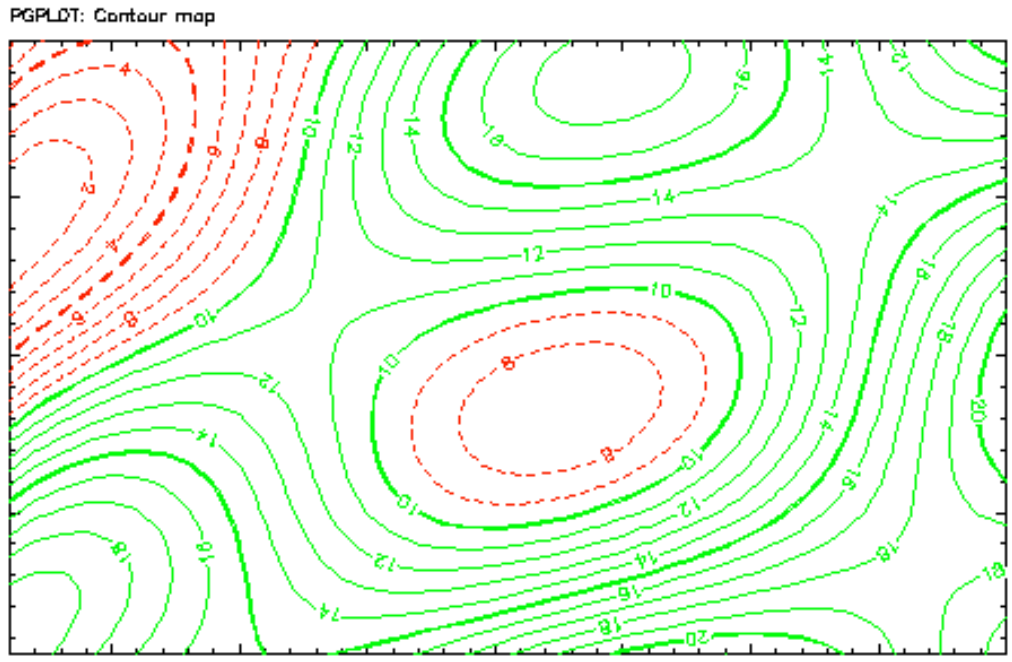
A histogram, e.g.



A scatter plot with both horizontal and vertical error bars:



A contour plot:



Note that the plots have ticks on all four sides (as required for publication) and that they have have titles (they should also have axes labeled, which seems optional in this demo I stole the screenshots from.

Also answer the following questions:

1. How difficult is the package to install? (easy, moderate, hard)
2. How many additional packages had to be installed? (0, 1, or many)
3. How hard was it to learn how to use?
4. How complete is the documentation? (good, OK, bad)
5. Can you get the cursor position from the package?
6. Other comments?

Test Results

VTK

The first package was VTK. This was attempted on Windows. The package was fairly easy to install, although a very large package ~ 20Mb + ~30Mb documentation. The package is heavily oriented toward 3D plotting. 2-D plotting seems to be possible, but the learning curve is very steep (users guide and design manuals are both proprietary ~$60) and 2-D plotting would use so little of VTK as to make us wonder why use it at all. The users guide and reference books do not explicitly cover 2-D plotting. Reports from users (T. Bridgeman in the Goddard SVS) say that the learning curve is indeed steep; furthermore, the software is free but the company makes money consulting with users helping them design applications. These considerations led us to drop any further evaluation of VTK as an exportable graphics environment.

The tests resulted in the following

Package	**Test/question**	**Result Linux**	**Result Windows**
Biggles (Python)	Histogram?	Yes	
	Scatter?	Yes	
	Contour?	Yes	
	Installation	Easy	
	Dependencies	3 (Numeric, math, and libplot)	
	Ease of Use	easy	
	Documentation	OK	
	Cursor position?	NO	

	Comments?	Very lightweight package. Understands TEX for labeling. More packages need to be downloaded if you want to do anything fancier. Nice sphere projection map comes with it. No cursor position available.	
plplot	Histogram	Yes	
	Scatter	Yes	
	Contour	Yes	
	Installation	Easy	
	Dependencies	Self-contained dist.	
	Ease of Use	Easy	
	Documentation	Good	
	Cursor Position	Yes	
	Comments	No Gif device available; JPEG and PNG device drivers seemed to be missing needed headers for libgd to compile form source – presumably we can distribute executables that only need to talk to gd. Python and C API.	
ROOT	Histogram	Yes	Yes
	Scatter	Yes	Yes
	Contour	Yes	Yes
	Installation	Easy	Easy
	Dependencies	(dist self contained)	Self-contained
	Ease of Use	moderate	Moderate
	Documentation	good	Good
	Cursor Position	Yes	Yes
	Comments	No Gif device came installed. Support page queried	Cannot save canvas jpeg or gif on Windows. Note: test was carried out by

			writing C++ code, not running from the ROOT command line.

Recommendations

The recommendation is of course related to what you want to do. As I see it there are three options to go with the three packages:

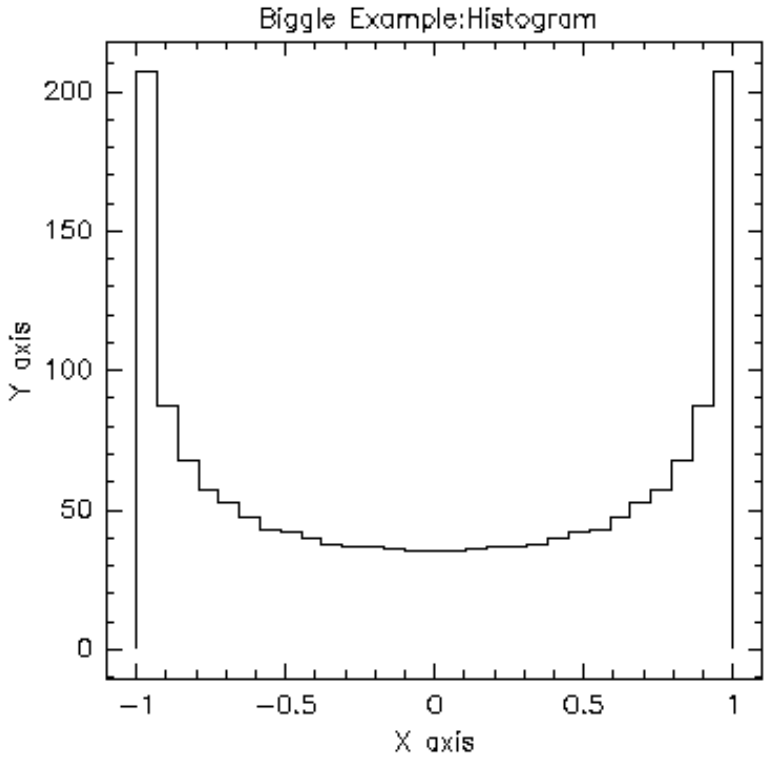
1) plotting done from script interface, and images would be handled by some other program, e.g., DS9. Here something like Biggles would work well for quick 2-D plots, the imaging and fancier plots could be done with the image processing tool.
2) Plotting thrown up by C++ and Python code + image processing tool: Here there is a direct interface to the compileable code. Any program can directly throw up plots of a reasonable publishable quality. Can make the same plots form the Python and C world. The easy lightweight solution wpi;d then be plplot.
3) Plotting from C++ and Python Code: If we want the GUIs, images, and Plots to be callable from C++ ROOT is our choice. If we want to keep scripting, we can do this with Python. http://proj-gaudi.web.cern.ch/proj-gaudi/RootPython/ has a Python wrapper for ROOT. The setup for RootPython was difficult and the interface proved to be fragile. Apparently later versions of Boost (after 1.25) have broken the RootPython package. How difficult will this be to maintain?

Recommendations are based on how we want to structure the User interface. If we want to just wrap C++ libraries in a scripting language and then use the scripting language to to handle graphics and GUIs, we probably don't need the full power of ROOT, and so Plplot would be a good choice. Plplot can also then be called from the C++ code.
If however we want the GUIs to be integrated into our C++ code, ROOT would be a good choice.

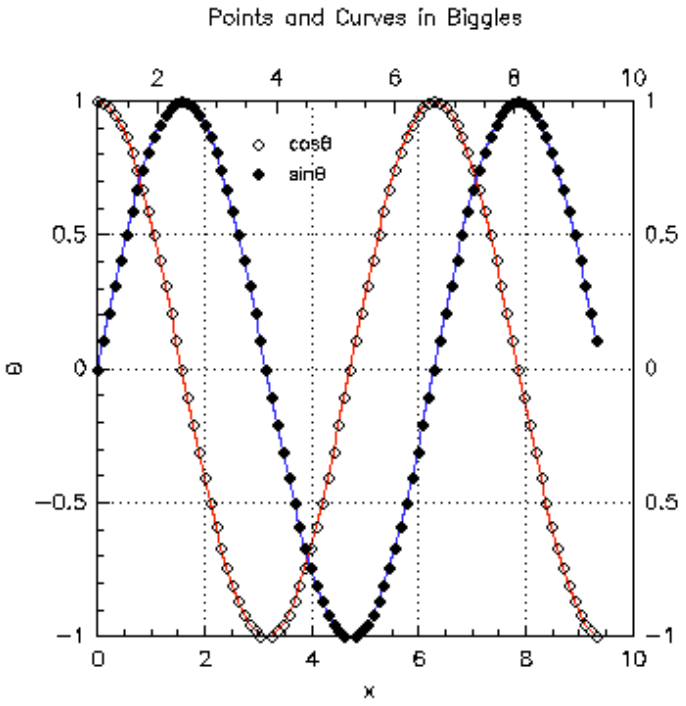
Appendix: Plot Shots

Biggles:

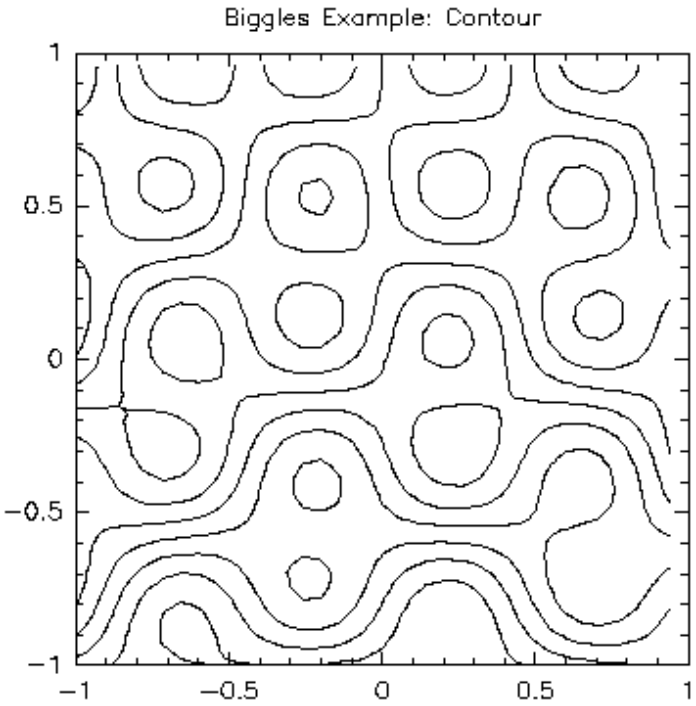
Histogram_Linux:



Scatter Plot:

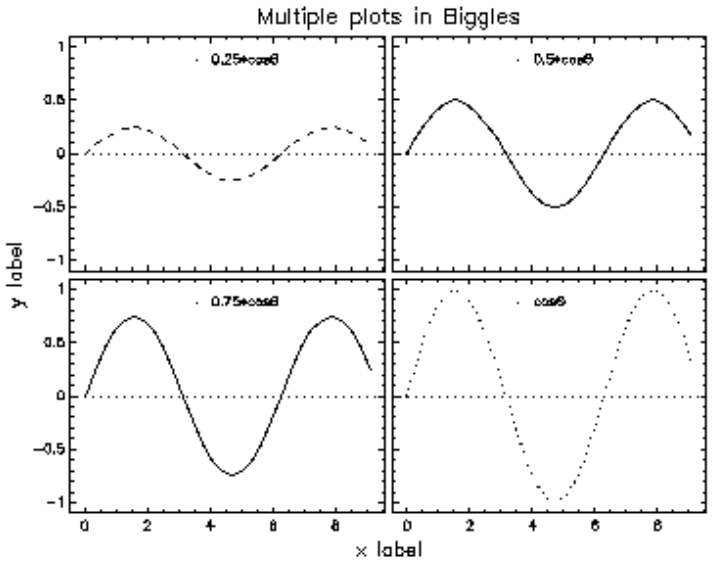


Linux Contour

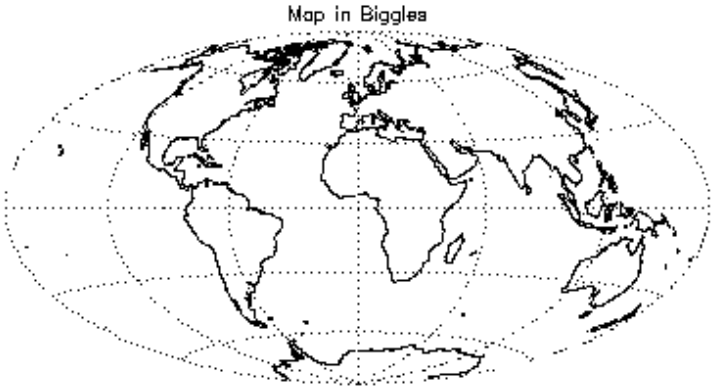


extras:

Linux mulitplot:

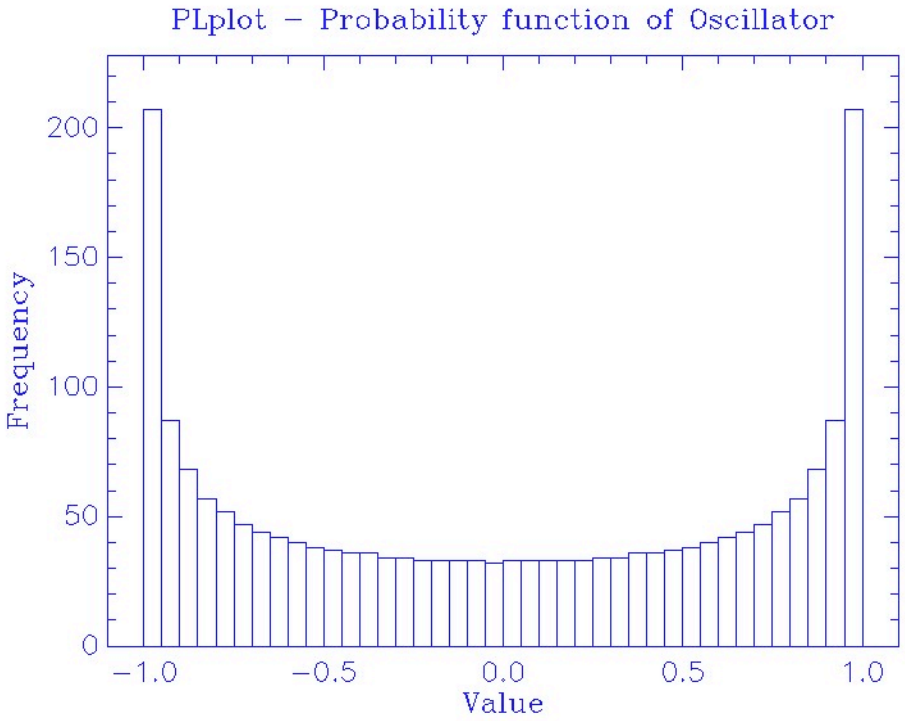


Linux Map:

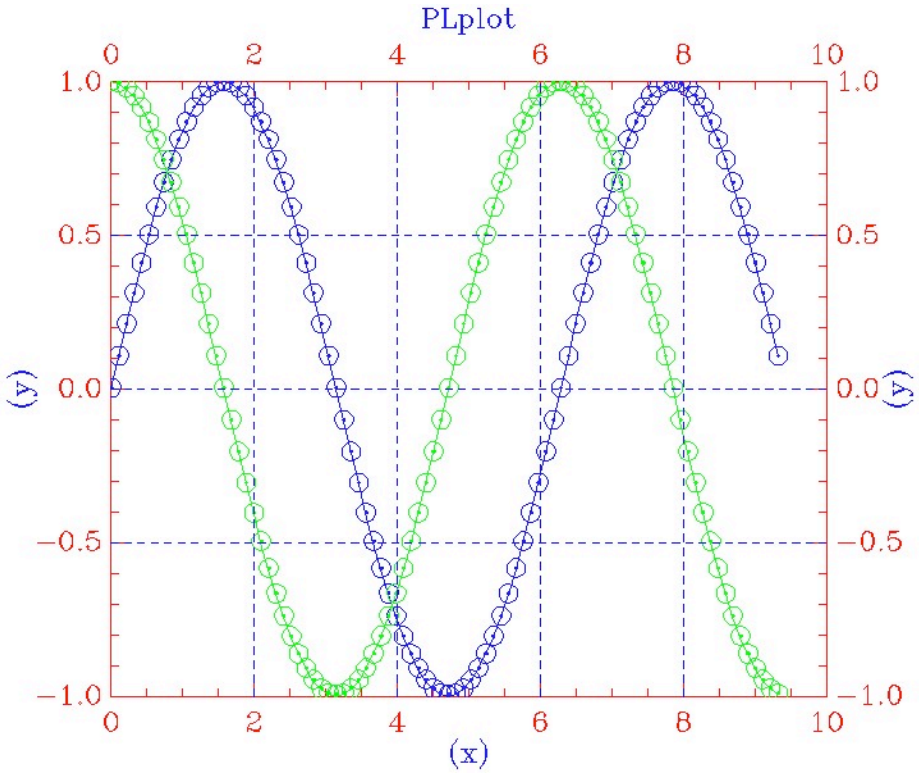


Plplot

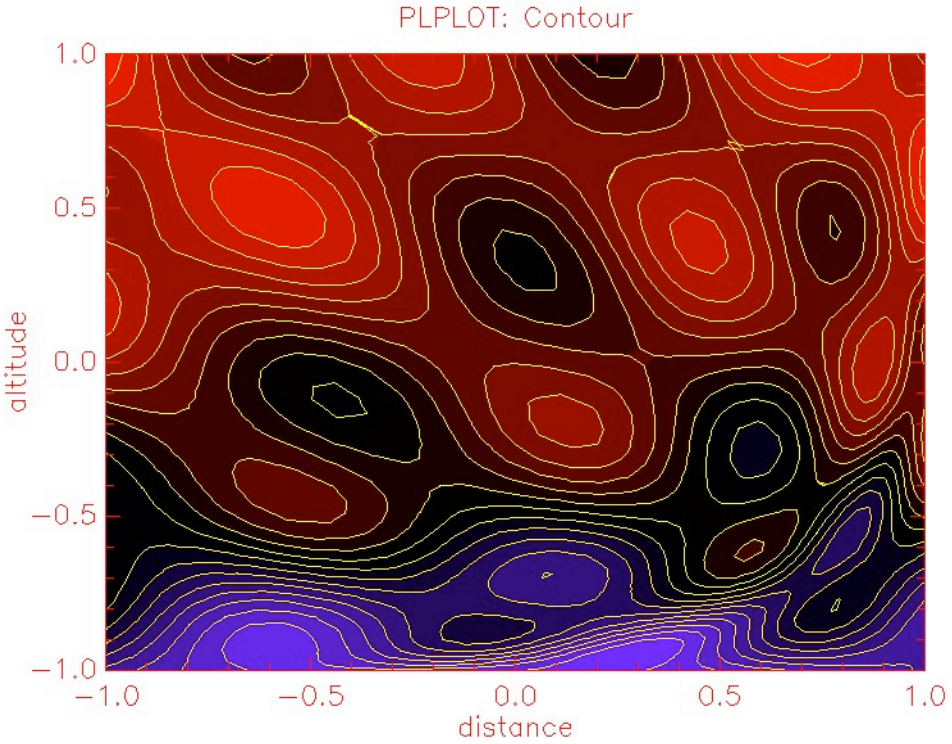
Linux Histogram:



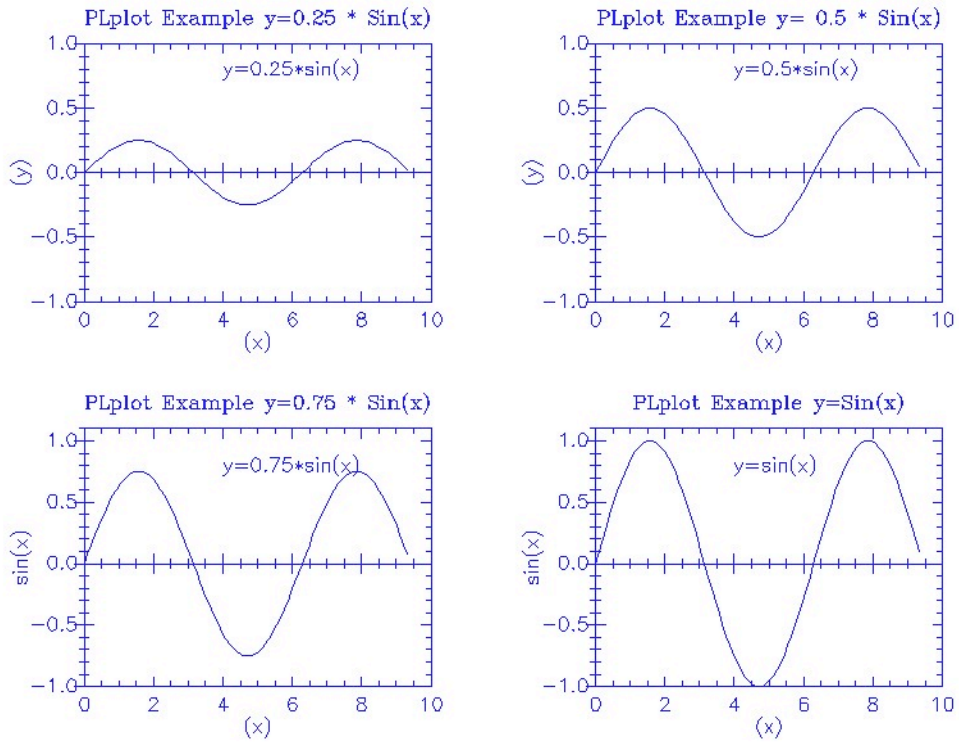
Linux Scatter:



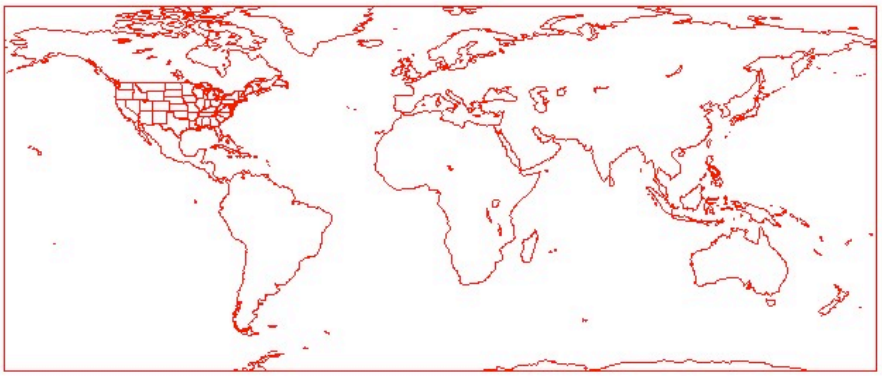
Linux contour



Linux Multi

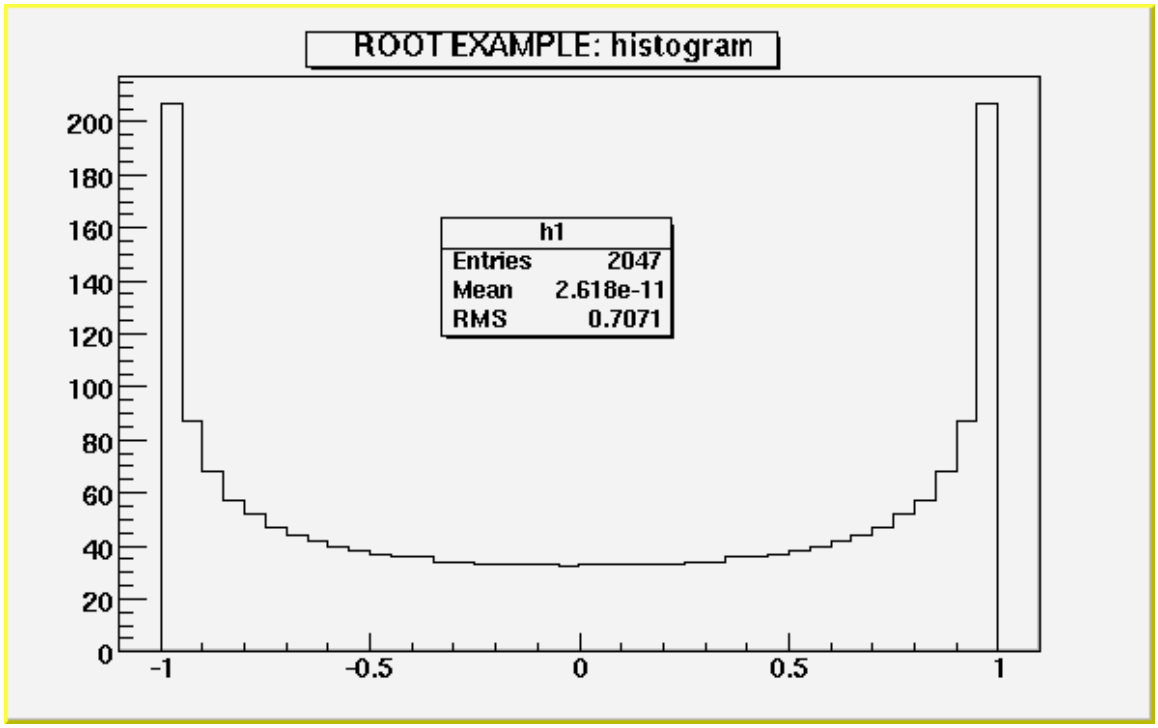


Linux map

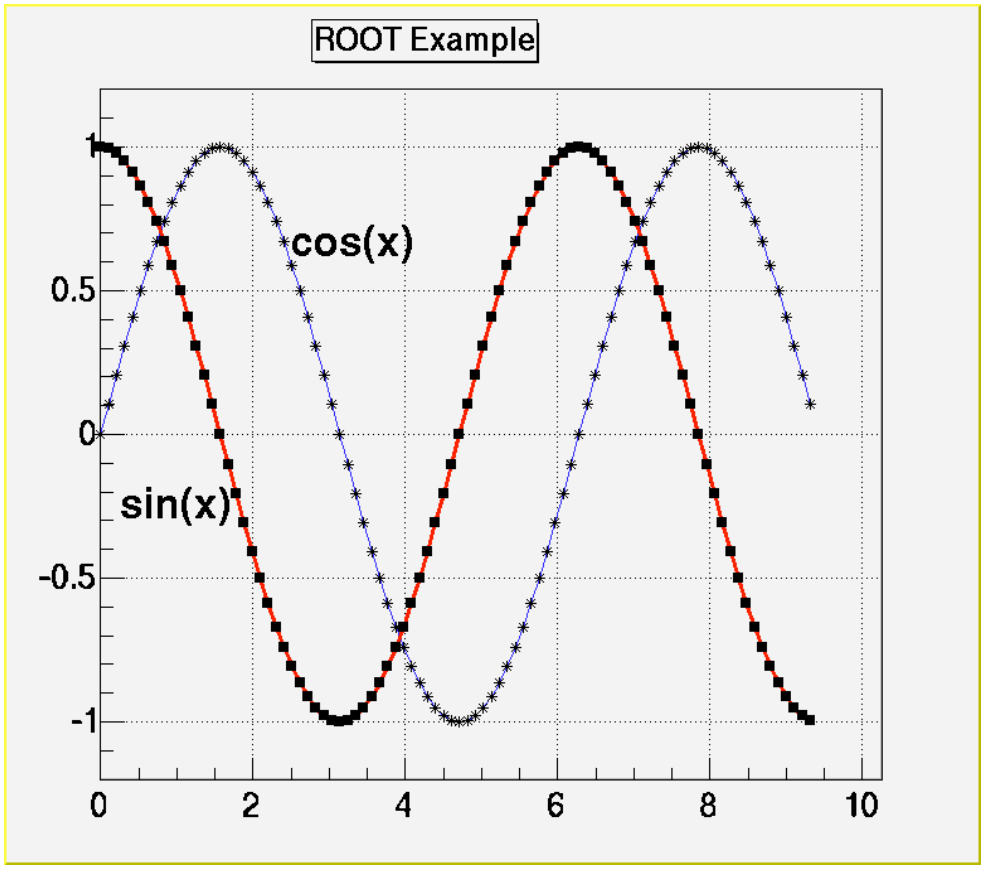


ROOT

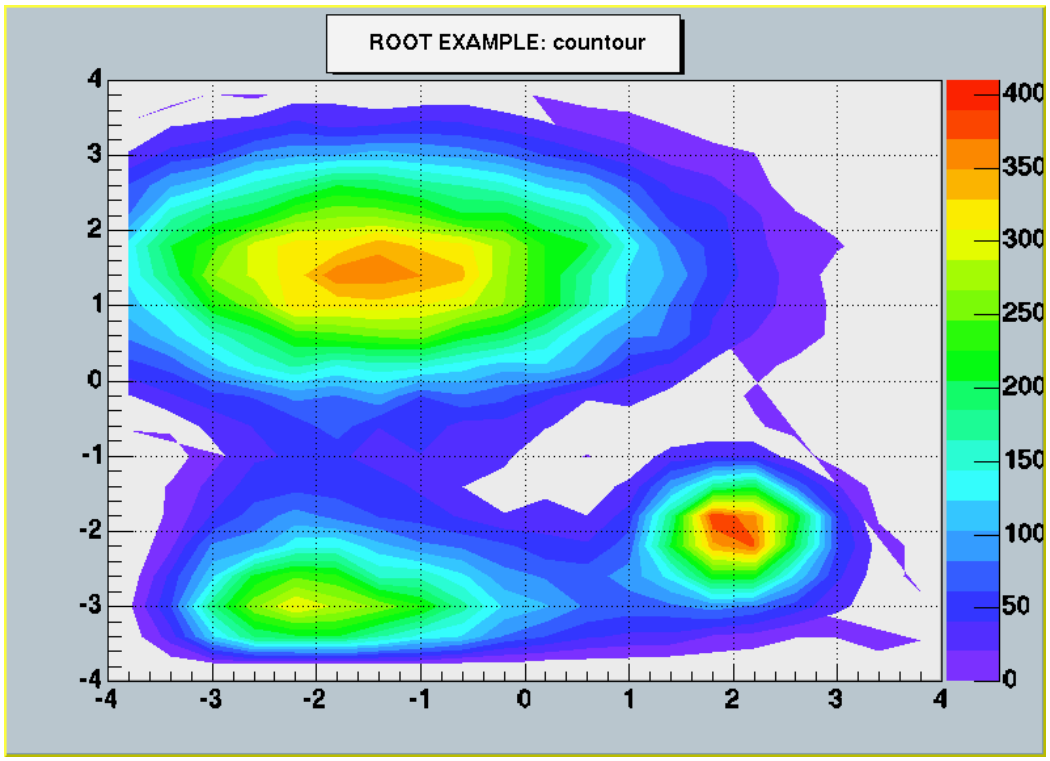
Linux Histogram



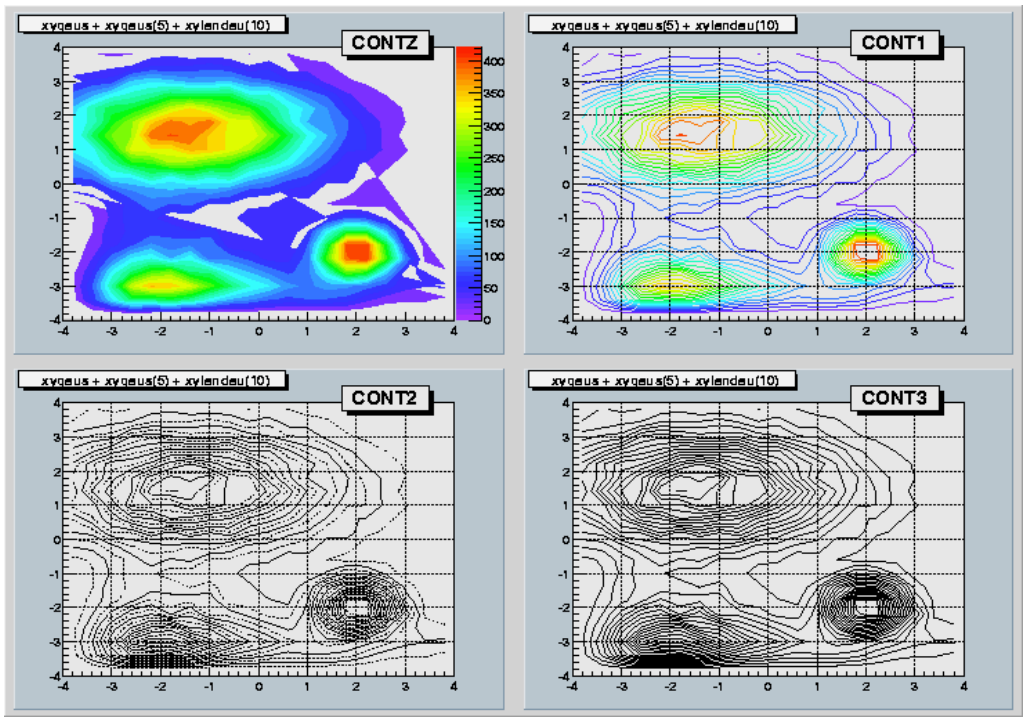
Linux Scatter plot



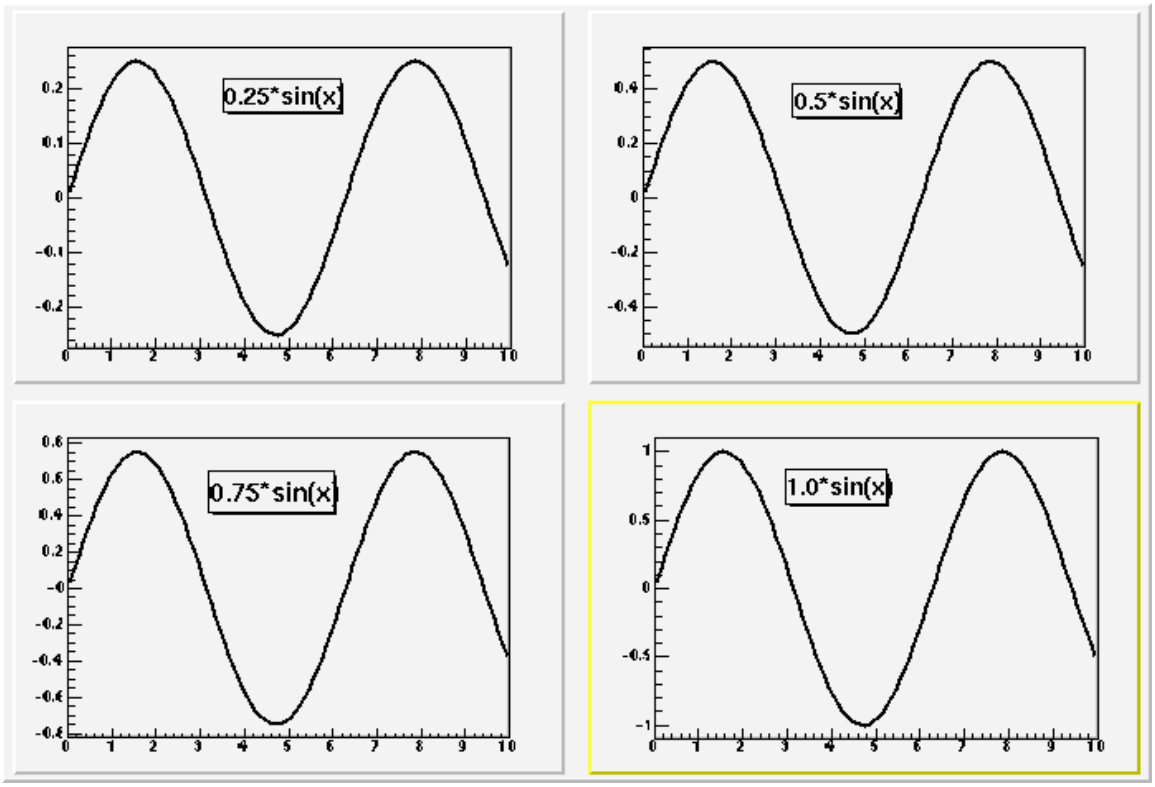
Linux Contour



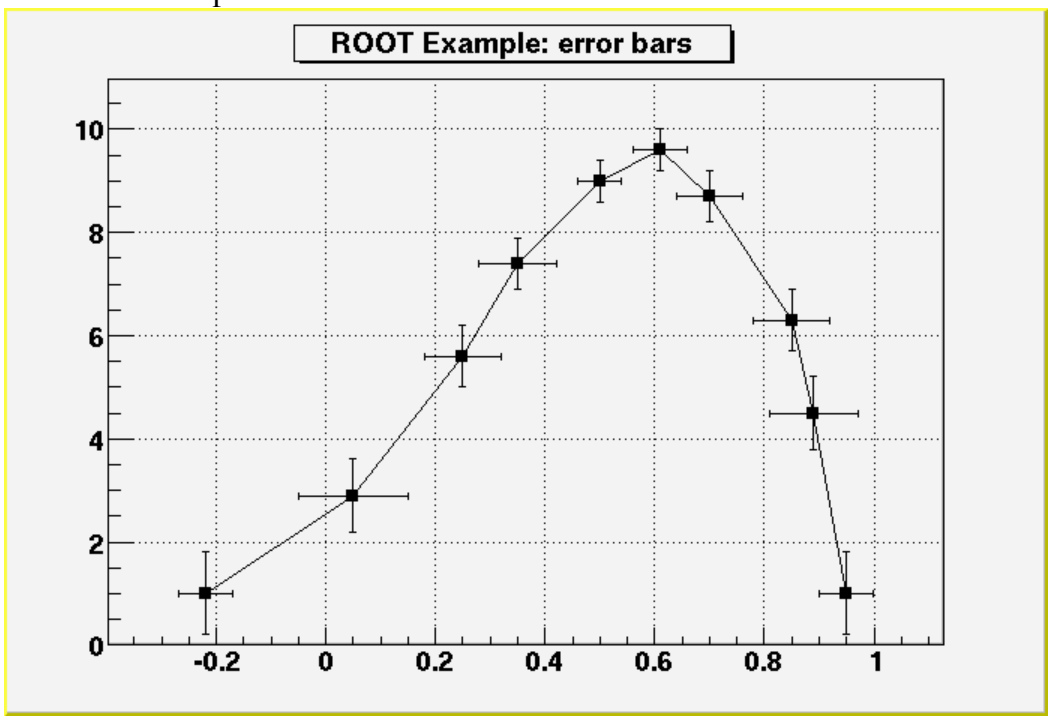
Windows contour/multi



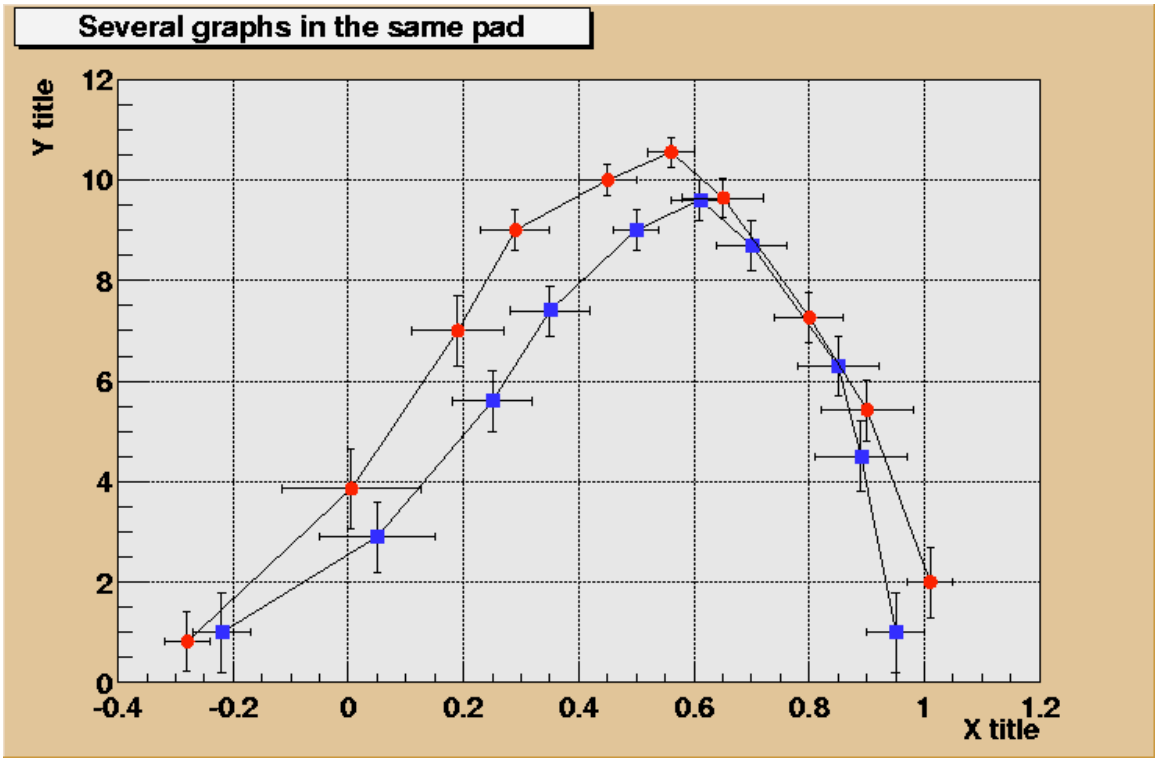
Linux Multi



Linux Error bar plot



Windows error bars



Linux_map: No obvious easy map display (maybe with INTEGRAL graphics libs)
Windows_map: Not attempted.